FOR PROJECT v0.2

Most of our functions will focus on

- **under user_management:**
  - settings.py
- **under users:**
  - forms.py
  - models.py
  - pdf_utils.py
  - urls.py
  - views.py

All our **css styles/UI** will be on **styles.css** *(Cambridge_teamProject/user_management/static/css)*

Filled forms, submitted forms, and signatures will store under media file
(Cambridge_teamProject/user_management/media)

How each HTML work:

Under user_mangement

- **home.html**: welcome page
- **base.html**: the overall outline of the web
- **dashboard.html**: dashboard page
- **form_selection.html**: page that select between 2 forms. You can download, upload, prefilled the forms
- **login.html**(under *Cambridge_teamProject/user_management/templates/socialaccount/login.html*): login page for Microsoft
- **login.html**(under *Cambridge_teamProject/user_management/templates/account/login.html*): login page
- **logout.html**: logout confirmation page
- **signup.html**: register page
- **form_status.html**: page to update each form's status like approve, pending, etc.
- **submitted_forms.html**: Page to view submitted forms

Under users

- **edit_profile.html**: modify your profile information
- **upload_signature.html**: upload your signature
- **user_confirm_delete.html**: user delete confirmation
- **user_form.html**: edit create update user
- **user_list.html**: user page where you can choose to edit, find, deactivate, etc.

HOW EACH FUNCTION WORKS

Under user_management

- **asgi.py:**
  - Lets Django project run on an **asynchronous server**.
  - Sets up Django and exposes the application object that servers like **Uvicorn** use to serve the app — especially useful for WebSockets or real-time apps.
- **settings.py:**
  - **`.env` support** via `load_dotenv()` — keeps sensitive info out of the codebase.
  - `SECRET_KEY`: Loaded from `.env`, fallback if missing.
  - `DEBUG`: Controlled via `.env` (defaults to `True`).
  - `ALLOWED_HOSTS`: Set to local dev (`127.0.0.1`, `localhost`).
  - Installed Apps:
    - Django built-ins (admin, auth, etc.)
    - users, allauth, bootstrap4, django_extensions
  - Middleware handles sessions, authentication, protection, and allauth-specific middleware (AccountMiddleware).
  - URL Routing & Templates
  - Database uses dj_database_url
  - Password Validation
  - Localization and time
  - Static and media files
  - Authentication
  - Microsoft Login Authentication - OAuth
- **urls.py:**
  - It routes URLs like /dashboard/, /users/, and /accounts/ to the right views.
  - It tells Django which views or apps should handle which URLs.
- **wsgi.py:**
  - Sets up the **WSGI application** that allows Django to run on **traditional web servers**
  - Entry point for deploying the project in a **production environment**.

Under users

- **admin.py:**
  - Customizes how CustomUser model appears in the admin.
  - It adds extra fields like role and status, makes sure the model always appears in the sidebar, and even renames the label to show it first.
  - By subclassing UserAdmin, you're modifying how user data is displayed and managed in the admin interface
- **apps.py:**
  - Sets the default primary key type and registers the app under the name "users"

- **forms.py:**
  - **Defines user-related forms** for creating users, uploading signatures, and editing profile information, with role-based field behavior.
  - class UserForm(forms.ModelForm):
    - A form for admins to create/update users.
    - It lets admins assign roles (basicuser or admin), but disables the role field if the logged-in user is not an admin.
  - class SignatureUploadForm(forms.ModelForm):
    - A form that lets users upload their **signature image** (mapped to the signature field in the CustomUser model).
  - class ProfileForm(forms.ModelForm):
    - Allows users to edit their **profile info** (phone number and student ID), and also updates first name, last name, and email from the user model.

- **models.py:**
  - **Defines all the core database models** for users, profiles, form submissions, and their version histories, enabling role-based access and PDF tracking.
  - class CustomUser(AbstractUser):
    - Extends user model with role, active status, signature image, and student ID.
  - class Profile(models.Model):
    - Stores additional user info like phone number and student ID, linked via one-to-one to CustomUser.
  - class SubmittedForm(models.Model):
    - Represents a user-submitted form with status tracking, comments, and a PDF file.
  - class SubmittedFormVersion(models.Model):
    - Stores version history of each SubmittedForm, including uploaded PDFs and optional approver signatures.
  - class FilledForm(models.Model):
    - Keeps track of individual form submissions (with version numbers) for user-filled PDFs separately from admin-reviewed versions.

- **pdf_utils.py:**
  - pdf_utils.py uses PyMuPDF to auto-fill PDF forms with user data and signature, then saves the filled PDF to the media directory. Opens a PDF file using PyMuPDF (fitz) and reads the first page.
  - Extracts user info (name, email, phone, student ID, current date) and maps it to likely field labels in the PDF.
  - Searches for each field label on the page, and writes the corresponding user data next to it.
  - Optionally embeds the user's signature image near a field labeled "Signature".
  - Saves the modified PDF to the media/filled_forms/ directory with a unique filename.
  - Returns the saved path so it can be stored or sent as a response.

- **urls.py:**
  - Maps URL paths to all user management, form submission, profile editing, and PDF-related views in the app.
  - Users:
    - ''user_list: List all users.
    - 'create/' → user_create: Add a new user.
    - '<int:pk>/edit/' → user_update: Edit an existing user.
    - '<int:pk>/delete/' → user_delete: Delete a user.
    - 'deactivate/<int:user_id>/' & 'reactivate/<int:user_id>/': Enable/disable user accounts.
  - Profile info & signature:
    - 'upload-signature/' → Upload user signature.
    - 'edit-profile/' → Edit user profile.
  - Forms:
    - 'forms/' → Show form selection page.
    - 'generate-form/<str:form_type>/' → Auto-fill and generate a form PDF.
    - 'submit-filled-pdf/<str:form_type>/' → Submit generated PDF.
    - 'upload-filled-pdf/<str:form_type>/' → Upload a hand-filled PDF.
    - 'submitted-forms/' → View all submitted forms.
    - 'view-submitted-form/<int:form_id>/' → View a specific PDF form file.
    - 'delete-submitted-form/<int:form_id>/' → Delete a submission.
    - 'update-form-status/<int:form_id>/' → Admin updates form status and version.
    - 'form-version/<int:version_id>/' → View an older version of a form.

- **views.py:**
  - Handles all user-related logic, including user management, profile and signature updates, form submission and versioning, PDF generation, and form approval workflows.
  - User Management Functions
    - user_list(request)
      - Displays a paginated, searchable, and sortable list of users (only for logged-in users).
    - user_create(request)
      - Allows admin users to create new users and assign roles.
    - user_update(request, pk)
      - Admins can update an existing user's info using a form.
    - user_delete(request, pk)
      - Admins can delete a user after confirmation.
    - deactivate_user(request, user_id)
      - Admins can deactivate a user account to prevent login.
    - reactivate_user(request, user_id)
      - Admins can reactivate a previously deactivated account.

- Profile and Signature Functions
  - upload_signature(request)
    - Users can upload their signature image, which will be used in PDF forms.
  - edit_profile(request)
    - Users can update their name, email, phone number, and student ID.
- Form Handling & PDF Generation
  - form_selection(request)
    - Shows available forms to fill, download, view, or upload.
  - generate_filled_pdf(request, form_type)
    - Auto-fills a blank PDF with the user's info and signature, and returns the filled file for download.
  - submit_filled_pdf(request, form_type)
    - Fills a PDF on the backend and submits it as a new SubmittedForm (only if not submitted before).
  - upload_filled_pdf(request, form_type)
    - Allows the user to upload a pre-filled PDF, saves it, and records a version in SubmittedFormVersion.
  - delete_submitted_form(request, form_id)
    - Deletes a previously submitted form and its file (either by user or superuser).
- Review & Approval
  - submitted_forms_list(request)
    - Shows a list of submitted forms. Admins see all; users see only their own.
  - form_status(request, form_id)
    - Admins can update the form's status (e.g., approve, return), add comments, and save a new version of the form.
- PDF Viewers
  - view_submitted_form(request, form_id)
    - Displays the most recent submitted form as a PDF. Also updates status from draft → pending.
  - view_form_version(request, version_id)
    - Opens a specific version of a submitted form PDF in a new tab for viewing.